

# t8code - modular adaptive mesh refinement in the exascale era

Johannes Holke<sup>1\*</sup>, Johannes Markert<sup>1\*</sup>, David Knapp<sup>1\*</sup>, Lukas Dreyer<sup>1</sup>, Sandro Elsweijer<sup>1</sup>, Niklas Böing<sup>1</sup>, Chiara Hergl<sup>1</sup>, Prasanna Ponnusamy<sup>1</sup>, and Achim Basermann<sup>1</sup>

<sup>1</sup> German Aerospace Center (DLR), Institute for Software Technology, Cologne, Germany  
Corresponding author \* These authors contributed equally.

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#)
- [Repository](#)
- [Archive](#)

Editor: [Open Journals](#)

## Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

## In partnership with



This article and software are linked with research article DOI [10.3847/xxxxx](https://doi.org/10.3847/xxxxx) <- [update this with the DOI from AAS once you know it.](#), published in the Astrophysical Journal <- The name of the AAS journal..

## Summary

In this note we present our software library t8code for scalable dynamic adaptive mesh refinement (AMR) officially released in 2022 ([Holke et al., 2022](#)). t8code is written in C/C++, open source, and readily available at [www.dlr-amr.github.io/t8code](https://www.dlr-amr.github.io/t8code). The library provides fast and memory efficient parallel algorithms for dynamic AMR to handle tasks such as mesh adaptation, load-balancing, ghost computation, feature search and more. t8code can manage meshes with over one trillion mesh elements ([Holke et al., 2021](#)) and scales up to one million parallel processes ([Holke, 2018](#)). It is intended to be used as mesh management back end in scientific and engineering simulation codes paving the way towards high-performance applications of the upcoming exascale era.

## Introduction

AMR has been established as a successful approach for scientific and engineering simulations over the past decades Babuška & Rheinboldt (1978). By modifying the mesh resolution locally according to problem specific indicators, the computational power is efficiently concentrated where needed and the overall memory usage is reduced by orders of magnitude. However, managing adaptive meshes and associated data is a very challenging task, especially for parallel codes. Implementing fast and scalable AMR routines generally leads to a large development overhead motivating the need for external mesh management libraries like t8code.

t8code is written in C/C++, open source, and the latest release can be obtained at <https://dlr-amr.github.io/t8code> ([Holke et al., 2022](#)). It uses efficient space-filling curves (SFC) to manage the data in structured refinement trees. While in the past being successfully applied to quadrilateral and hexahedral meshes Weinzierl (2019), t8code extends these SFC techniques in a modular fashion, such that arbitrary element shapes are supported. We achieve this modularity through a novel decoupling approach that separates high-level (mesh global) algorithms from low-level (element local) implementations. All high-level algorithms can then be applied to different implementations of element shapes and refinement patterns. A mix of different element shapes in the same mesh is also supported.

Currently, t8code provides implementations of Morton type SFCs with  $1 : 2^d$  refinement for vertices ( $d = 0$ ), lines ( $d = 1$ ), quadrilaterals, triangles ( $d = 2$ ), hexahedra, tetrahedra, prisms, and pyramids ( $d = 3$ ). The latter having a  $1 : 10$  refinement rule with tetrahedra emerging as child elements (Knapp, 2020). Additionally, implementation of other refinement patterns and SFCs is possible according to the specific requirements of the application.

The purpose of this note is to provide a brief overview and a first point of entrance for software

41 developers working on codes storing data on (distributed) meshes.

42 For further information beyond this short note and also for code examples, we refer to our  
43 Documentation and Wiki (Holke et al., 2022) and our other technical papers on t8code Lilikakis  
44 (2022).

## 45 References

- 46 Babuška, I., & Rheinboldt, W. C. (1978). Error estimates for adaptive finite element  
47 computations. *SIAM Journal on Numerical Analysis*, 15(4), 736–754. <https://doi.org/10.1137/0715049>  
48
- 49 Bangerth, W., Hartmann, R., & Kanschat, G. (2007). Deal.II—a general-purpose object-  
50 oriented finite element library. *ACM Trans. Math. Softw.*, 33(4), 24–es. <https://doi.org/10.1145/1268776.1268779>  
51
- 52 Becker, F. (2021). *Removing hanging faces from tree-based adaptive meshes for numerical*  
53 *simulations* [Master's thesis]. Universität zu Köln.
- 54 Burstedde, C., & Holke, J. (2016). A tetrahedral space-filling curve for nonconforming adaptive  
55 meshes. *SIAM Journal on Scientific Computing*, 38, C471–C503. <https://doi.org/10.1137/15M1040049>  
56
- 57 Burstedde, C., & Holke, J. (2017). Coarse Mesh Partitioning for Tree-Based AMR. *SIAM Jour-*  
58 *nal on Scientific Computing*, Vol. 39, C364–C392. <https://doi.org/10.1137/16M1103518>
- 59 Burstedde, C., Wilcox, L. C., & Ghattas, O. (2011). p4est: Scalable Algorithms for Parallel  
60 Adaptive Mesh Refinement on Forests of Octrees. *SIAM Journal on Scientific Computing*,  
61 33(3), 1103–1133. <https://doi.org/10.1137/100791634>
- 62 Dörfler, W. (1996). A convergent adaptive algorithm for poisson's equation. *SIAM Journal on*  
63 *Numerical Analysis*, 33(3), 1106–1124. <https://doi.org/10.1137/0733054>
- 64 Dreyer, L. (2021). *The local discontinuous galerkin method for the advection-diffusion*  
65 *equation on adaptive meshes* [Master's thesis, Rheinische Friedrich-Wilhelms-Universität  
66 Bonn]. <https://elib.dlr.de/143969/>
- 67 Elswijker, S. (2021). *Curved Domain Adaptive Mesh Refinement with Hexahedra*. Hochschule  
68 Bonn-Rhein-Sieg. <https://elib.dlr.de/143537/>
- 69 Holke, J. (2018). *Scalable algorithms for parallel tree-based adaptive mesh refinement with*  
70 *general element types* [{PhD} thesis]. Rheinische Friedrich-Wilhelms-Universität Bonn.
- 71 Holke, J., Burstedde, C., Knapp, D., Dreyer, L., Elswijker, S., Uenlue, V., Markert, J., Lilikakis,  
72 I., & Boeing, N. (2022). *t8code* (Version 1.0.0). <https://doi.org/10.5281/zenodo.7034838>
- 73 Holke, J., Knapp, D., & Burstedde, C. (2021). An Optimized, Parallel Computation of the  
74 Ghost Layer for Adaptive Hybrid Forest Meshes. *SIAM Journal on Scientific Computing*,  
75 C359–C385. <https://doi.org/10.1137/20M1383033>
- 76 Knapp, D. (2020). *A space-filling curve for pyramidal adaptive mesh refinement* [{M}aster's  
77 Thesis]. Rheinische Friedrich-Wilhelms-Universität Bonn.
- 78 Lilikakis, I. (2022). *Algorithms for tree-based adaptive meshes with incomplete trees* [Master's  
79 thesis, Universität zu Köln]. <https://elib.dlr.de/191968/>
- 80 Teunissen, J., & Keppens, R. (2019). A geometric multigrid library for quadtree/octree  
81 AMR grids coupled to MPI-AMRVAC. *Computer Physics Communications*, 245, 106866.  
82 <https://doi.org/https://doi.org/10.1016/j.cpc.2019.106866>

- <sup>83</sup> Weinzierl, T. (2019). The Peano Software-Parallel, Automaton-based, Dynamically Adaptive  
<sup>84</sup> Grid Traversals. *ACM Transactions on Mathematical Software*, 45(2), 1–41. [https:](https://doi.org/10.1145/3319797)  
<sup>85</sup> [//doi.org/10.1145/3319797](https://doi.org/10.1145/3319797)

DRAFT